# Álgebra Relacional



# Linguagens de Consulta Relacionais

- Linguagens de consulta (*query languages*): permitem recuperar e manipular dados.
- O modelo relacional suporta linguagens de consulta simples e poderosas:
  - Embasamento formal sólido.
  - Espaço para otimização automatizada.
- Não são linguagens de programação!
  - Não são "computacionalmente completas".
  - Inadequadas para cálculos complexos.
  - Suportam acesso fácil e eficiente a BDs.

# Linguagens de Consulta Formais

Duas linguagens de consulta matemáticas formam a base para a definição e para a implementação de linguagens “reais” (isto é, SQL):

❶ Álgebra relacional: mais operacional, útil para representar planos de execução de consultas.

❷ Cálculo relacional: permite que o usuário especifique o que deseja, sem dizer como o sistema deve proceder.

☞ Conhecimentos de álgebra relacional são fundamentais para se entender SQL!




# Álgebra Relacional

- Operações básicas:
  - Seleção ($\sigma$) - Seleciona um subconjunto de linhas de uma relação.
  - Projeção ($\Pi$) - Remove colunas não desejadas.
  - Produto cartesiano ($\times$) - Combina duas relações.
  - Diferença de conjuntos (–) - Tuplas na primeira relação mas não na segunda.
  - União ($\cup$) - Tuplas na primeira relação mais tuplas na segunda (tuplas que pertencem à primeira relação ou à segunda).

# Álgebra Relacional (cont.)

- Operações adicionais:
  (Não essenciais, porém muito úteis!)
  – Intersecção ($\cap$)
  – Junção ou *join* ($\bowtie$)
  – Divisão ($\div$)
  – Operação de renomear tabela ou atributo ($\rho$)
- Como cada operação retorna uma relação, pode-se fazer composição de operações. (A álgebra é “fechada”.)




# Exemplos de Tabelas

- Estas instâncias das relações “Pilotos” e “Reservas” serão usadas nos exemplos adiante.

P1

| num_p | nome_p | nível | idade |
|---|---|---|---|
| 22 | Joca | 7 | 45 |
| 31 | Laerte | 8 | 55 |
| 58 | Rafa | 10 | 35 |

P2

| num_p | nome_p | nível | idade |
|---|---|---|---|
| 28 | Zeca | 9 | 35 |
| 31 | Laerte | 8 | 55 |
| 44 | Giba | 5 | 35 |
| 58 | Rafa | 10 | 35 |

R1

| num_p | num_a | data |
|---|---|---|
| 22 | 101 | 10/10/98 |
| 58 | 103 | 11/12/98 |

## Projeção: $\Pi_{\text{lista\_de\_atribs}}$(relação)

- Remove os atributos que não estão na lista de atributos.
- Esquema do resultado contém exatamente os atributos listados.
  - O operador $\Pi$ deve eliminar repetições! (Por quê?)
  - Sistemas reais só eliminam repetições se o usuário pedir.

| nome_p | nível |
|---|---|
| Zeca | 9 |
| Laerte | 8 |
| Giba | 5 |
| Rafa | 10 |

$\Pi_{\text{nome\_p, nível}}$(P2)

| idade |
|---|
| 35 |
| 55 |

$\Pi_{\text{idade}}$(P2)




## Seleção: $\sigma_{\text{cond}}$(relação)

- Seleciona linhas que satisfazem a condição.
- Resultado sem linhas repetidas. (Por quê?)
- O esquema do resultado é o mesmo da relação de entrada.
- A relação resultante pode ser a entrada para outra operação (composição).

| num_p | nome_p | nível | idade |
|---|---|---|---|
| 28 | Zeca | 9 | 35 |
| 58 | Rafa | 10 | 35 |

$\sigma_{\text{nível} > 8}$(P2)

| nome_p | nível |
|---|---|
| Zeca | 9 |
| Rafa | 10 |

$\Pi_{\text{nome\_p, nível}}(\sigma_{\text{nível} > 8}(\text{P2}))$

## União, Intersecção, Diferença de Conjuntos

- Tomam como entrada duas relações, que precisam ser <u>compatíveis para união</u>:
  – Mesmo número de colunas.
  – Colunas 'correspondentes' com o mesmo tipo.
- Qual o esquema do resultado?

| num_p | nome_p | nível | idade |
|---|---|---|---|
| 22 | Joca | 7 | 45 |
| 31 | Laerte | 8 | 55 |
| 58 | Rafa | 10 | 35 |
| 44 | Giba | 5 | 35 |
| 28 | Zeca | 9 | 35 |

P1 ∪ P2

| num_p | nome_p | nível | idade |
|---|---|---|---|
| 22 | Joca | 7 | 45 |

P1 – P2

| num_p | nome_p | nível | idade |
|---|---|---|---|
| 31 | Laerte | 8 | 55 |
| 58 | Rafa | 10 | 35 |

P1 ∩ P2




## Produto Cartesiano

- P1 × R1: Cada linha de P1 é emparelhada com cada linha de R1.
- Esquema resultante tem os atributos de P1 e os de R1, com nomes 'herdados' se possível.

| (num_p) | nome_p | nível | idade | (num_p) | num_a | data |
|---|---|---|---|---|---|---|
| 22 | Joca | 7 | 45 | 22 | 101 | 10/10/98 |
| 22 | Joca | 7 | 45 | 58 | 103 | 11/12/98 |
| 31 | Laerte | 8 | 55 | 22 | 101 | 10/10/98 |
| 31 | Laerte | 8 | 55 | 58 | 103 | 11/12/98 |
| 58 | Rafa | 10 | 35 | 22 | 101 | 10/10/98 |
| 58 | Rafa | 10 | 35 | 58 | 103 | 11/12/98 |

Conflito: Tanto P1 como R1 tem uma coluna num_p.

<u>Renomear</u>: $\rho(C(1 \rightarrow num\_p\_1,\ 5 \rightarrow num\_p\_2),\ P1 \times R1)$

# Junções (*Joins*)

- <u>Junção condicional</u>: $R \bowtie_{cond} S = \sigma_{cond}(R \times S)$

| (num_p) | nome_p | nível | idade | (num_p) | num_a | data |
|---|---|---|---|---|---|---|
| 22 | Joca | 7 | 45 | 58 | 103 | 11/12/98 |
| 31 | Laerte | 8 | 55 | 58 | 103 | 11/12/98 |

$$P1 \bowtie_{P1.num\_p < R1.num\_p} R1$$

  - Esquema do resultado igual ao do produto cartesiano.
  - Menos linhas que o produto cartesiano, pode ser computado mais eficientemente.
  - Algumas vezes chamado de <u>junção teta</u> (*theta-join*).




# Junções (*Joins*)

- <u>Equijunção</u>: Caso especial de junção condicional, no qual a condição só contém igualdades de atributos.

| num_p | nome_p | nível | idade | num_a | data |
|---|---|---|---|---|---|
| 22 | Joca | 7 | 45 | 101 | 10/10/98 |
| 58 | Rafa | 10 | 35 | 103 | 11/12/98 |

$$P1 \bowtie_{num\_p} R1$$

  - Esquema do resultado como o do produto cartesiano, mas com uma só cópia das colunas para as quais foi especificada a igualdade.

☞ <u>Junção natural</u> (*natural join*): Equijunção sobre todos os atributos comuns às duas relações.